Particle-in-cell simulations

Part III: Boundary conditions and parallelization

Benoît Cerutti

IPAG, CNRS, Université Grenoble Alpes, Grenoble, France.

Plan of the lectures

• **Monday:**
  - *Morning*: The PIC method, numerical schemes and main algorithms.
  - *Afternoon*: Coding practice of the Boris push and the Yee algorithm.

• **Tuesday:**
  - *Morning*: Implementation of Zeltron, structure and methods.
  - *Afternoon*: Zeltron hands on relativistic reconnection simulations
  - *Evening*: Seminar about application of PIC to pulsar magnetospheres.

• **Wednesday:**
  - *Morning*: Boundary conditions and parallelization in Zeltron.
  - *Afternoon*: Zeltron Hands on relativistic collisionless shocks simulations
Field boundary conditions: a few examples

Periodic

Perfectly conducting walls

Absorbing layer (open boundary)

Absorbing layer

\[ \frac{\partial E}{\partial t} + \sigma E = c \nabla \times B - 4 \pi J \]

\[ \frac{\partial B}{\partial t} + \sigma \ast B = -c \nabla \times E \]
Example of a 1D absorbing layer

Absorption without reflection => **Gradually increasing** conductivity
For example:

\[ \sigma = -\sigma_0 (x - x_0)^3 \]

\(t=0,\) Gaussian pulse

No reflection!
Perfectly Matched Layer (PML)

\[
\begin{align*}
\frac{\partial E}{\partial t} + \sigma E &= c \nabla \times B - 4\pi J \\
\frac{\partial B}{\partial t} + \sigma^* B &= -c \nabla \times E
\end{align*}
\]

Multi-D generalization: Perfectly Matched Layer (see Bérenger 1994-1996)

**Example:** Let's consider a 2D case in vacuum with \(E_x, E_y\), and \(B_z\).

Then, we have to solve these:

\[
\begin{align*}
\frac{\partial E_y}{\partial t} &= -c \frac{\partial B_z}{\partial x} \quad \rightarrow \quad \text{Wave along x} \\
\frac{\partial E_x}{\partial t} &= c \frac{\partial B_z}{\partial y} \quad \rightarrow \quad \text{Wave along y} \\
\frac{\partial B_z}{\partial t} &= -c \left( \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} \right) \quad \rightarrow \quad \text{Wave along x and y}
\end{align*}
\]

The trick is to split the \(B_z\) component into two: \(B_z = B_{zx} + B_{zy}\)

\[
\begin{align*}
\frac{\partial E_y}{\partial t} + \sigma_x E_y &= -c \frac{\partial}{\partial x} (B_{zx} + B_{zy}) \\
\frac{\partial E_x}{\partial t} + \sigma_y E_x &= c \frac{\partial}{\partial y} (B_{zx} + B_{zy}) \\
\frac{\partial B_{zx}}{\partial t} + \sigma_x B_{zx} &= -c \frac{\partial E_y}{\partial x} \\
\frac{\partial B_{zy}}{\partial t} + \sigma_y B_{zy} &= c \frac{\partial E_x}{\partial y}
\end{align*}
\]

Easily generalized to all components in 2D and 3D.

Problem: 2 times more equations to solve!
Field boundary conditions in Zeltron

Choice of boundary conditions (*mod_input.f90*)

```fortran
! Specify the boundary conditions for the fields:
! 1. "PERIODIC": Periodic boundary conditions
! 2. "METAL": Perfect metal with infinite conductivity

CHARACTER(LEN=10), PARAMETER, PUBLIC :: BOUND_FIELD_XMIN="PERIODIC"
CHARACTER(LEN=10), PARAMETER, PUBLIC :: BOUND_FIELD_XMAX="PERIODIC"
CHARACTER(LEN=10), PARAMETER, PUBLIC :: BOUND_FIELD_YMIN="PERIODIC"
CHARACTER(LEN=10), PARAMETER, PUBLIC :: BOUND_FIELD_YMAX="PERIODIC"
```

Perfectly conducting wall along x-direction for $E_z$ (*mod_fields.f90*)

```fortran
!***********************************************************************
! Check boundary conditions along X

IF (xminp.EQ.xmin) THEN
  IF (BOUND_FIELD_XMIN.EQ."METAL") THEN
    ! Tangent to conductor surface
    Ez(1,:) = 0.0
  END IF
END IF

!***********************************************************************
```
Particle boundary conditions: a few examples

Periodic

Perfectly reflective walls

Injection absorption

\( \text{Ex: At } x=L \)

Positions:

\[
\begin{align*}
x & \leftarrow 2L - x \\
y & \leftarrow y
\end{align*}
\]

Velocities:

\[
\begin{align*}
v_x & \leftarrow -v_x \\
v_y & \leftarrow v_y
\end{align*}
\]
Particle boundary conditions in Zeltron

Choice of boundary conditions (mod_input.f90)

! Specify the boundary conditions for the particles:
! 1. "PERIODIC": Periodic boundary conditions
! 2. "REFLECT": Particles are elastically reflected at the wall
! 3. "ABSORB": Particles are absorbed at the wall

CHARACTER(LEN=10), PARAMETER, PUBLIC :: BOUND_PART_XMIN="PERIODIC"
CHARACTER(LEN=10), PARAMETER, PUBLIC :: BOUND_PART_XMAX="PERIODIC"
CHARACTER(LEN=10), PARAMETER, PUBLIC :: BOUND_PART_YMIN="PERIODIC"
CHARACTER(LEN=10), PARAMETER, PUBLIC :: BOUND_PART_YMAX="PERIODIC"

Chunk from SUBROUTINE BOUNDARIES_PARTICLES (mod_particles.f90)

!***********************************************************************
! Case 1: x>xmax
!***********************************************************************

IF (x.GT.xmax) THEN

! Elastic reflection
IF (BOUND_PART_XMAX.EQ."REFLECT") THEN
x=2.0*xmax-x
ux=-ux
END IF

! Absorption
IF (BOUND_PART_XMAX.EQ."ABSORB") THEN
wt=0d0
END IF

END IF

B. Cerutti
Parallelization: Domain decomposition

PIC code are really demanding in computing resources => **Need to parallelize the code!**

A common practice is to use the **Message Passing Interface (MPI)** library and the domain decomposition technique.

**Example:** Consider a 2D mesh 9x9 cells and 9 CPUs.

1D decomposition

2D decomposition

Applicable to an **arbitrary number of CPUs**
Choice decomposition depends on the problem
Define a topology

**SUBROUTINE COM_TOPOLOGY** in *mod_initial.f90*

```fortran
! Initialization of the cartesian topology
periods(1)=.TRUE.
periods(2)=.TRUE.
reorder=.FALSE.
dims(1)=NPX ! Number of processors along X
dims(2)=NPY ! Number of processors along Y

! Creation of the dimension in each direction
CALL MPI_DIMS_CREATE(NPROC,2,dims,ierr)

! Creation of the topology
CALL MPI_CART_CREATE(MPI_COMM_WORLD,2,dims,periods,reorder,COMM,ierr)

! To obtain the ID number of each process
CALL MPI_COMM_RANK(COMM,id,ierr)

! To obtain the coordinates of the process
CALL MPI_CART_COORDS(COMM,id,2,coords,ierr)
```

**2D decomposition**

```
  id=6   id=7   id=8
  id=3   id=4   id=5
  id=0   id=1   id=2
```
Local grids and arrays

Each processor has its own **local grid** and **local particle arrays** (*main.f90*)

```fortran
! Spatial boundaries in the X-direction of each domain
DOUBLE PRECISION :: xminp, xmaxp

! Spatial boundaries in the Y-direction of each domain
DOUBLE PRECISION :: yminp, ymaxp

! Global nodal grid
DOUBLE PRECISION, DIMENSION(1:NX) :: xg
DOUBLE PRECISION, DIMENSION(1:NY) :: yg
! Nodal grid in each domain
DOUBLE PRECISION, DIMENSION(1:NXP) :: xgp
DOUBLE PRECISION, DIMENSION(1:NYP) :: ygp

! Yee grid in each domain
DOUBLE PRECISION, DIMENSION(1:NXP) :: xyeep
DOUBLE PRECISION, DIMENSION(1:NYP) :: yyeep

! SPATIAL BOUNDARIES FOR EACH SUB-DOMAIN
xminp=xmin+coords(1)*NCXP*dx
xmaxp=xminp+NCXP*dx

yminp=ymin+coords(2)*NCYP*dy
ymaxp=yminp+NCYP*dy
```

B. Cerutti
Once the topology defined, it is crucial that each processor knows its **neighbours**

In Zeltron this information is contained in:

- **1D decomposition**
  - L id R

- **2D decomposition**
  - NW N NE
  - W id E
  - SW S SE

```plaintext
! ngh: neighbor array (1D)
INTEGER, DIMENSION(2) :: ngh

! ngh: neighbor array (2D)
INTEGER, DIMENSION(8) :: ngh

! ngh: neighbor array (3D)
INTEGER, DIMENSION(26) :: ngh
```
Communications between CPUs: Fields

Example: We want to compute E field on the grid (SUBROUTINE FIELDS_NODES in mod_fields.f90)

\[ Ex_{i,j} = \frac{Ex_{i+1/2,j} + Ex_{i-1/2,j}}{2} \]

But we need \( Ex_{-1/2,j} \) to compute \( Ex_{0,j} \)

This value is known by the neighbour \( W \)

\[ => W \text{ must send its values of } Ex_{nxp-1,j} \]
Communications between CPUs: Fields

A very typical MPI "point-to-point" communication of a 1D array in Zeltron (from \textit{mod\_fields.f90})

\begin{verbatim}
ALLOCATE(bufS2(1:NYP),bufR2(1:NYP))

bufS2=Ex(NXP-1,:) 

IF (MOD(id,2).EQ.0) THEN
  ! For even CPU id
  CALL MPI_SENDRECV(bufS2,NYP,MPI_DOUBLE_PRECISION,ngh(2),tag2,&
                    bufR2,NYP,MPI_DOUBLE_PRECISION,ngh(4),tag2,COMM,stat,ierr)
ELSE
  ! For odd CPU id
  CALL MPI_SENDRECV(bufS2,NYP,MPI_DOUBLE_PRECISION,ngh(2),tag2,&
                    bufR2,NYP,MPI_DOUBLE_PRECISION,ngh(4),tag2,COMM,stat,ierr)
ENDIF
\end{verbatim}
Communications between CPUs: Particles

MPI Communications

1D: Up to 2 / CPU

2D: Up to 8 / CPU

3D: Up to 26 / CPU

Example: 2D decomposition

B. Cerutti
Communications between CPUs: Particles

Steps for exchanging particles

SUBROUTINE COM_PARTICLES (mod_motion.f90)

Step 1: Count all particles leaving the processor domain towards the neighbouring processors.

Step 2: Ask the neighbours how many particles are leaving their domains towards processor id.
Communications between CPUs: Particles

**Step 3:** Exchange particle data \((x, y, z, ux, uy, uz, wgt, \text{tag}, \ldots)\)

<table>
<thead>
<tr>
<th>Neighbour W</th>
<th>id</th>
<th>Neighbour E</th>
</tr>
</thead>
<tbody>
<tr>
<td>[...]</td>
<td>[...]</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{bufRW} \Rightarrow \text{pcl}[N_{\text{inc}}(W)] \quad \text{bufSE} \Rightarrow \text{pcl}[N_{\text{esc}}(E)]
\]

CALL MPI_SENDRECV(bufSE, NESC(2) * 11, MPI_DOUBLE PRECISION, ngh(2), tag2, &
bufRW, NINC(4) * 11, MPI_DOUBLE PRECISION, ngh(4), tag2, COMM, stat, ierr)

<table>
<thead>
<tr>
<th>Neighbour W</th>
<th>id</th>
<th>Neighbour E</th>
</tr>
</thead>
<tbody>
<tr>
<td>[...]</td>
<td>[...]</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{bufSW} \Rightarrow \text{pcl}[N_{\text{esc}}(W)] \quad \text{bufSE} \Rightarrow \text{pcl}[N_{\text{inc}}(E)]
\]

CALL MPI_SENDRECV(bufSW, NESC(4) * 11, MPI_DOUBLE PRECISION, ngh(4), tag4, &
bufRE, NINC(2) * 11, MPI_DOUBLE PRECISION, ngh(2), tag4, COMM, stat, ierr)

**Step 4:** Resize particle array to update the content of particles in each domain.

\[
pcl(N_{new}) \leftarrow pcl(N_{old} - N_{esc} + N_{inc})
\]

B. Cerutti
PIC codes scale well to large number of CPUs

The era of **High-Performance Computing!** Today ~> \(10^6\) CPUs

See [http://www.top500.org/](http://www.top500.org/)

**OSIRIS Code**

**Zeltron Code**

---

B. Cerutti
Load balancing issues

Computing time (without communications): ~90% particles, ~10% fields

Many particles Processor #5 is slowing down all the others

Few particles Processor #9 is waiting for all the others

B. Cerutti
A specific example: a reconnecting layer

Density contrast $\sim 10!$

Some solutions:
- Appropriate domain decomposition
- Dynamical changes of the decomposition
- Varying particle weights
- Hybrid code: MPI-OpenMP

B. Cerutti
Another specific example: a shock

Density contrast $\sim 4$

**High**-particle density
Downstream, shocked flow

**Low**-particle density
Upstream, unshocked plasma

1D decomposition is appropriate here, but maximum number of cores is **limited**.

B. Cerutti
Hybrid parallelization: MPI-OpenMP

Supercomputer

Node 1  Node 2  Node 3  ...  ...  Node N

Network (InfiniBand)

Example: 2 nodes, 4 processors per node. 1D decomposition

• Pure MPI:

Memory is shared within a node

The particle loop can be parallelized with OpenMP within a node.
=> Bigger domain, better load balancing.

• MPI-OpenMP:

Node 1

Node 2

B. Cerutti
**Hands-on III: Relativistic collisionless shocks**

Collisionless shock sounds counter-intuitive. To form a shock we need collisions, something that thermalizes the flow (randomize particle's velocity). In collisionless shocks, waves and magnetic irregularities effectively collide with particles.

**Main astrophysical applications:**

**Gamma-ray bursts**

**Pulsar Wind Nebulae**

- External shock (afterglow)
- Wind termination shock

**Relativistic jets**

Lobes, collimation, internal shocks

How efficient at accelerating particles?
What are the main acceleration mechanisms?
The usual numerical setups

Simulation frame = Frame of the contact discontinuity

**Receding wall at V=c**

**Injection of cold plasma**

**Conducting reflective wall**

**Reflected flow** $V=-c$

**Shock**

$\Gamma \gg 1$

Good setup to follow the formation of one shock only (the reverse shock)

Simulation frame = Frame of the downstream flow

**Receding wall at V=c**

**Injection of cold plasma**

**Conducting reflective wall**

**Target plasma at rest**

**Periodic**

$\Gamma \gg 1$

Good setup to follow the formation of all the shocks plus contact discontinuity
Unmagnetized collisionless shock formation

**Phase 1:** The two flows overlap without interacting

**Phase 2:** Electromagnetic counter-streaming instabilities grows (linear phase)

**Phase 3:** Non-linear phase, the shock form and particle acceleration begins

Unshocked "upstream"  Shocked "downstream"

Precursor  Shock front $\sim c/3$

Instabilities

Density

1  2  3

Rankine-Hugoniot

25